Binary Classification / Perceptron

Nicholas Ruozzi
University of Texas at Dallas

Slides adapted from David Sontag and Vibhav Gogate
Reminders

• Homework 1 is available on eLearning and due in 2 weeks
  • Late homework will not be accepted

• Instructions for getting started with the course, e.g., joining Piazza & taking the preq quiz, are on eLearning

• Office hours are happening this week
  • Prof. (blackboard) T 3:45pm-4:45pm, W 11:00am-12:00pm
Supervised Learning

• **Input:** \((x^{(1)}, y^{(1)}), \ldots, (x^{(M)}, y^{(M)})\)

  • \(x^{(m)}\) is the \(m^{th}\) data item and \(y^{(m)}\) is the \(m^{th}\) label

• **Goal:** find a function \(f\) such that \(f(x^{(m)})\) is a “good approximation” to \(y^{(m)}\)

  • Can use it to predict \(y\) values for previously unseen \(x\) values
Supervised Learning

• **Hypothesis space**: set of allowable functions $f: X \rightarrow Y$

• **Goal**: find the “best” element of the hypothesis space

  • How do we measure the quality of $f$?
Examples of Supervised Learning

• Spam email detection
• Handwritten digit recognition
• Stock market prediction
• More?
Regression

Hypothesis class: linear functions $f(x) = ax + b$

How do we measure the quality of the approximation?
Linear Regression

• In typical regression applications, measure the fit using a squared loss function

\[ L(f) = \frac{1}{M} \sum_m (f(x^{(m)}) - y^{(m)})^2 \]

• Want to minimize the average loss on the training data

• For 2-D linear regression, the learning problem is then

\[ \min_{a,b} \frac{1}{M} \sum_m (ax^{(m)} + b - y^{(m)})^2 \]

• For an unseen data point, \( x \), the learning algorithm predicts \( f(x) \)
Supervised Learning

- **Select a hypothesis space** (elements of the space are represented by a collection of parameters)
- **Choose a loss function** (evaluates quality of the hypothesis as a function of its parameters)
- **Minimize loss function, e.g., using gradient descent** (minimization over the parameters)
- **Evaluate quality of the learned model using test data** – that is, data on which the model was not trained
Binary Classification

- Regression operates over a continuous set of outcomes
- Suppose that we want to learn a function $f: X \to \{0,1\}$
- As an example:

<table>
<thead>
<tr>
<th></th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

How do we pick the hypothesis space?

How do we find the best $f$ in this space?
Binary Classification

• Regression operates over a continuous set of outcomes
• Suppose that we want to learn a function \( f: X \rightarrow \{0,1\} \)
• As an example:

<table>
<thead>
<tr>
<th></th>
<th>(x_1)</th>
<th>(x_2)</th>
<th>(x_3)</th>
<th>(y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

How many functions with three binary inputs and one binary output are there?
Binary Classification

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>?</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>?</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>?</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>?</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

$2^8$ possible functions

$2^4$ are consistent with the observations

How do we choose the best one?

What if the observations are noisy?
Challenges in ML

• How to choose the right hypothesis space?
  • Number of factors influence this decision: difficulty of learning over the chosen space, how expressive the space is, ...

• How to evaluate the quality of our learned hypothesis?
  • Prefer “simpler” hypotheses (to prevent overfitting)
  • Want the outcome of learning to generalize to unseen data
Binary Classification

- Input \((x^{(1)}, y^{(1)}), \ldots, (x^{(M)}, y^{(M)})\) with \(x^{(m)} \in \mathbb{R}^n\) and \(y^{(m)} \in \{-1, +1\}\)

- We can think of the observations as points in \(\mathbb{R}^n\) with an associated sign (either +/- corresponding to 0/1)

- An example with \(n = 2\)
Binary Classification

- Input \((x^{(1)}, y^{(1)}), ..., (x^{(M)}, y^{(M)})\) with \(x^{(m)} \in \mathbb{R}^n\) and \(y^{(m)} \in \{-1, +1\}\)

- We can think of the observations as points in \(\mathbb{R}^n\) with an associated sign (either +/- corresponding to 0/1)

- An example with \(n = 2\)

What is a good hypothesis space for this problem?
Binary Classification

• Input \((x^{(1)}, y^{(1)}), ..., (x^{(M)}, y^{(M)})\) with \(x^{(m)} \in \mathbb{R}^n\) and \(y^{(m)} \in \{-1, +1\}\)

• We can think of the observations as points in \(\mathbb{R}^n\) with an associated sign (either +/- corresponding to 0/1)

• An example with \(n = 2\)

What is a good hypothesis space for this problem?
Binary Classification

• Input \((x^{(1)}, y^{(1)}), \ldots, (x^{(M)}, y^{(M)})\) with \(x^{(m)} \in \mathbb{R}^n\) and \(y^{(m)} \in \{-1, +1\}\)

• We can think of the observations as points in \(\mathbb{R}^n\) with an associated sign (either +/- corresponding to 0/1)

• An example with \(n = 2\)

In this case, we say that the observations are linearly separable
Linear Separators

• In \( n \) dimensions, a hyperplane is a solution to the equation

\[
 w^T x + b = 0
\]

with \( w \in \mathbb{R}^n, b \in \mathbb{R} \)

• Hyperplanes divide \( \mathbb{R}^n \) into two distinct sets of points (called open halfspaces)

\[
 w^T x + b > 0
\]

\[
 w^T x + b < 0
\]
Binary Classification

- Input \((x^{(1)}, y^{(1)}), ..., (x^{(M)}, y^{(M)})\) with \(x^{(m)} \in \mathbb{R}^n\) and \(y^{(m)} \in \{-1, +1\}\)

- We can think of the observations as points in \(\mathbb{R}^n\) with an associated sign (either +/- corresponding to 0/1)

- An example with \(n = 2\)

In this case, we say that the observations are linearly separable.
The Linearly Separable Case

• Input \((x^{(1)}, y^{(1)}), \ldots, (x^{(M)}, y^{(M)})\) with \(x^{(m)} \in \mathbb{R}^n\) and \(y^{(m)} \in \{-1, +1\}\)

• Hypothesis space: separating hyperplanes

\[ f(x) = \text{sign} \ (w^T x + b) \]

• How should we choose the loss function?
The Linearly Separable Case

- Input \((x^{(1)}, y^{(1)}), \ldots, (x^{(M)}, y^{(M)})\) with \(x^{(m)} \in \mathbb{R}^n\) and \(y^{(m)} \in \{-1, +1\}\)

- Hypothesis space: separating hyperplanes

\[
f(x) = \text{sign} (w^T x + b)
\]

- How should we choose the loss function?

  - Count the number of misclassifications

\[
loss = \sum_m |y^{(m)} - \text{sign}(w^T x^{(m)} + b)|
\]

- Tough to optimize, gradient contains no information
The Linearly Separable Case

• Input \((x^{(1)}, y^{(1)}), \ldots, (x^{(M)}, y^{(M)})\) with \(x^{(m)} \in \mathbb{R}^n\) and \(y^{(m)} \in \{-1, +1\}\)

• Hypothesis space: separating hyperplanes

\[ f(x) = \text{sign} (w^T x + b) \]

• How should we choose the loss function?

  • Penalize misclassification linearly by the size of the violation

\[ \text{perceptron loss} = \sum_m \max\{0, -y^{(m)} (w^T x^{(m)} + b)\} \]

  • Modified hinge loss (this loss is convex, but not differentiable)
The Perceptron Algorithm

• Try to minimize the perceptron loss using gradient descent

  • The perceptron loss isn't differentiable, how can we apply gradient descent?

  • Need a generalization of what it means to be the gradient of a convex function
Gradients of Convex Functions

- For a differentiable convex function $g(x)$ its gradients are **linear underestimators**
Gradients of Convex Functions

- For a differentiable convex function $g(x)$ its gradients are **linear underestimators**.
Gradients of Convex Functions

• For a differentiable convex function $g(x)$ its gradients are \textbf{linear underestimators}: zero gradient corresponds to a global optimum
Subgradients

• For a convex function $g(x)$, a subgradient at a point $x^0$ is given by any line, $l$, such that $l(x^0) = g(x^0)$ and $l(x) \leq g(x)$ for all $x$, i.e., it is a linear underestimator.
Subgradients

• For a convex function $g(x)$, a subgradient at a point $x^0$ is given by any line, $l$, such that $l(x^0) = g(x^0)$ and $l(x) \leq g(x)$ for all $x$, i.e., it is a linear underestimator.
Subgradients

- For a convex function $g(x)$, a subgradient at a point $x^0$ is given by any line, $l$, such that $l(x^0) = g(x^0)$ and $l(x) \leq g(x)$ for all $x$, i.e., it is a linear underestimate.
Subgradients

- For a convex function $g(x)$, a subgradient at a point $x^0$ is given by any line, $l$, such that $l(x^0) = g(x^0)$ and $l(x) \leq g(x)$ for all $x$, i.e., it is a linear underestimator.

If $\vec{0}$ is a subgradient at $x^0$, then $x^0$ is a global minimum.
Subgradients

- If a convex function is differentiable at a point $x$, then it has a unique subgradient at the point $x$ given by the gradient.

- If a convex function is not differentiable at a point $x$, it can have many subgradients.

  - E.g., the set of subgradients of the convex function $|x|$ at the point $x = 0$ is given by the set of slopes $[-1,1]$.

- Subgradients only guaranteed to exist for convex functions.
The Perceptron Algorithm

• Try to minimize the perceptron loss using (sub)gradient descent
The Perceptron Algorithm

- Try to minimize the perceptron loss using (sub)gradient descent

\[
\nabla_w (\text{perceptron loss}) = - \sum_{m=1}^{M} \left( y^{(m)} x^{(m)} \cdot 1_{-y^{(m)} f_{w,b}(x^{(m)}) \geq 0} \right)
\]

\[
\nabla_b (\text{perceptron loss}) = - \sum_{m=1}^{M} \left( y^{(m)} \cdot 1_{-y^{(m)} f_{w,b}(x^{(m)}) \geq 0} \right)
\]
The Perceptron Algorithm

- Try to minimize the perceptron loss using (sub)gradient descent

\[
\nabla_w \text{(perceptron loss)} = - \sum_{m=1}^{M} \left( y^{(m)} x^{(m)} \cdot 1_{-y^{(m)}f_w,b(x^{(m)}) \geq 0} \right)
\]

\[
\nabla_b \text{(perceptron loss)} = - \sum_{m=1}^{M} \left( y^{(m)} \cdot 1_{-y^{(m)}f_w,b(x^{(m)}) \geq 0} \right)
\]

Is equal to zero if the \(m^{th}\) data point is correctly classified and one otherwise.
The Perceptron Algorithm

• Try to minimize the perceptron loss using (sub)gradient descent

\[ w^{(t+1)} = w^{(t)} + \gamma_t \sum_{m=1}^{M} \left( y^{(m)} x^{(m)} \cdot 1 - y^{(m)} f_{w,b}(x^{(m)}) \geq 0 \right) \]

\[ b^{(t+1)} = b^{(t)} + \gamma_t \sum_{m=1}^{M} \left( y^{(m)} \cdot 1 - y^{(m)} f_{w,b}(x^{(m)}) \geq 0 \right) \]

• With step size \( \gamma_t \) (also called the learning rate)

• Note that, for convergence of subgradient methods, a diminishing step size, e.g., \( \gamma_t = \frac{1}{1+t} \) is required
Stochastic Gradient Descent

• To make the training more practical, stochastic (sub)gradient descent is often used instead of standard gradient descent

• Approximate the gradient of a sum by sampling a few indices (as few as one) uniformly at random and averaging

$$\nabla_x \left[ \sum_{m=1}^{M} g_m(x) \right] \approx \frac{1}{K} \sum_{k=1}^{K} \nabla_x g_{m_k}(x)$$

here, each $m_k$ is sampled uniformly at random from $\{1, \ldots, M\}$

• Stochastic gradient descent converges to the global optimum under certain assumptions on the step size
Stochastic Gradient Descent

• Setting $K = 1$, we pick a random observation $m$ and perform the following update

if the $m^{th}$ data point is misclassified:

\[
\begin{align*}
    w^{(t+1)} &= w^{(t)} + \gamma_t y^{(m)} x^{(m)} \\
    b^{(t+1)} &= b^{(t)} + \gamma_t y^{(m)}
\end{align*}
\]

if the $m^{th}$ data point is correctly classified:

\[
\begin{align*}
    w^{(t+1)} &= w^{(t)} \\
    b^{(t+1)} &= b^{(t)}
\end{align*}
\]

• Sometimes, you will see the perceptron algorithm specified with $\gamma_t = 1$ for all $t$
Applications of Perceptron

• Spam email classification
  • Represent emails as vectors of counts of certain words (e.g., sir, madam, Nigerian, prince, money, etc.)
  • Apply the perceptron algorithm to the resulting vectors
  • To predict the label of an unseen email
    • Construct its vector representation, $x'$
    • Check whether or not $w^T x' + b$ is positive or negative
Perceptron Learning Drawbacks

• No convergence guarantees if the observations are not linearly separable

• Can overfit

  • There can be a number of perfect classifiers, but the perceptron algorithm doesn’t have any mechanism for choosing between them
What If the Data Isn’t Separable?

- Input \((x^{(1)}, y^{(1)}), ..., (x^{(M)}, y^{(M)})\) with \(x^{(m)} \in \mathbb{R}^n\) and \(y^{(m)} \in \{-1, +1\}\)
- We can think of the observations as points in \(\mathbb{R}^n\) with an associated sign (either +/- corresponding to 0/1)
- An example with \(n = 2\)

What is a good hypothesis space for this problem?
What If the Data Isn’t Separable?

- Input \((x^{(1)}, y^{(1)}), \ldots, (x^{(M)}, y^{(M)})\) with \(x^{(m)} \in \mathbb{R}^n\) and \(y^{(m)} \in \{-1, +1\}\)

- We can think of the observations as points in \(\mathbb{R}^n\) with an associated sign (either +/- corresponding to 0/1)

- An example with \(n = 2\)

What is a good hypothesis space for this problem?
Adding Features

• Perceptron algorithm only works for linearly separable data

Can add **features** to make the data linearly separable in a higher dimensional space!

Essentially the same as higher order polynomials for linear regression!
Adding Features

• The idea, choose a feature map $\phi: \mathbb{R}^n \rightarrow \mathbb{R}^k$
  
  • Given the observations $x^{(1)}, \ldots, x^{(M)}$, construct feature vectors $\phi(x^{(1)}), \ldots, \phi(x^{(M)})$
  
  • Use $\phi(x^{(1)}), \ldots, \phi(x^{(M)})$ instead of $x^{(1)}, \ldots, x^{(M)}$ in the learning algorithm
  
  • Goal is to choose $\phi$ so that $\phi(x^{(1)}), \ldots, \phi(x^{(M)})$ are linearly separable in $\mathbb{R}^k$
  
  • Learn linear separators of the form $w^T \phi(x)$ (instead of $w^T x$)

• Warning: more expressive features can lead to overfitting!
Adding Features: Examples

- $\phi \left( \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \right) = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

  - This is just the input data, without modification

- $\phi \left( \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \right) = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_1^2 \\ x_2^2 \end{bmatrix}$

  - This corresponds to a second degree polynomial separator, or equivalently, elliptical separators in the original space
Adding Features

\[(x_1 - 1)^2 + (x_2 - 1)^2 - 1 \leq 0\]
Adding Features

\[ 1x_1^2 + 1x_2^2 - 2x_1 - 2x_2 - 2 \leq 0 \]
Support Vector Machines

• How can we decide between two perfect classifiers?

• What is the practical difference between these two solutions?